[Data Ingestion] --> [Multimodal Embedding Storage (FAISS + Vector DB)] --> [Query Processing (LLM + Retrieval)] --> [UI (Streamlit)]

↑

[Multimodal Fusion (text + image encoder)]

**📊 Your Model Comparison Results**

**BLIP Model**

* ✅ **Embedding**: 512 dimensions
* ✅ **Caption Generated**: "a book cover with a blue background and a black background"
* ✅ **Unique Feature**: Only model that can describe images in words

**CLIP Model**

* ✅ **Embedding**: 512 dimensions
* ✅ **Same embedding values as BLIP** (they share the same CLIP backbone)
* ✅ **Fast loading** (already downloaded)

**ViT Model**

* ✅ **Embedding**: 768 dimensions (largest feature space)
* ✅ **Different embedding pattern** (notice the larger values: -0.11, 0.56, -0.55)
* ⚠️ **Slower download** (346MB model file)
* ⚠️ **Warning about training** (but still works for embeddings)

**ResNet Model**

* ✅ **Embedding**: 2048 dimensions (highest dimensional)
* ✅ **Sparse embeddings** (many zeros: 0.0, 0.0, 0.01...)
* ✅ **Classic CNN architecture**

**🏆 For Your News RAG System - Recommendation:**

**Choose CLIP** because:

1. **✅ Perfect for News RAG**:
   * Handles both text queries AND images
   * Can match "AI breakthrough" text to robot images
   * Trained on diverse web content (like news)
2. **✅ Practical Advantages**:
   * Fast loading (shared with BLIP)
   * Reasonable embedding size (512 vs 2048)
   * Proven performance on multimodal tasks
3. **✅ Your Test Shows**:
   * Clean, normalized embeddings
   * No errors or issues
   * Ready to use immediately

**📝 For Your Project Report:**

markdown

## Model Selection Process

Tested 4 vision models on identical news image:

| Model | Embedding Size | Key Strengths | Test Result |

|-------|---------------|---------------|-------------|

| CLIP | 512 | Text-image alignment | ✅ Fast, clean embeddings |

| ViT | 768 | Pure visual understanding | ✅ Works but slower download |

| ResNet | 2048 | Classic CNN features | ✅ Sparse but functional |

| BLIP | 512 | Caption generation | ✅ Bonus: generated description |

**\*\*Selected: CLIP\*\*** - Best balance of speed, size, and multimodal capability for news retrieval.

**Your testing worked perfectly! Now you can confidently use CLIP for your news RAG system.** 🎯

**1. Усі моделі успішно завантажуються**

* Значення loading\_success: true для всіх енкодерів означає, що **інтеграція з Hugging Face працює правильно**.
* Це підтверджує, що середовище налаштоване коректно (права доступу, PyTorch, transformers, timm, huggingface\_hub тощо працюють стабільно).

**⏱ 2. Час завантаження суттєво відрізняється**

| **Модель** | **Час завантаження (с)** |
| --- | --- |
| CLIP | 1.5 |
| ResNet | 4.0 |
| ViT | 9.5 |
| BLIP | 40.6 |

* **BLIP** — найдовше завантажується (майже 1 хвилина), бо важить майже 1 GB і включає як текстовий, так і візуальний енкодер.
* **CLIP** — найшвидший серед мультимодальних моделей.

**Висновок:** якщо потрібна швидкість — краще використовувати CLIP або ResNet.

**💾 3. Ресурсомісткість**

| **Модель** | **Розмір (MB)** | **Вектор (embedding)** |
| --- | --- | --- |
| CLIP | 500 | 512 |
| ViT | 441 | 768 |
| ResNet | 102 | 2048 |
| BLIP | 990 | 512 |

* **ResNet** дає найдовший вектор (2048), хоча сам файл — відносно легкий (102MB).
* **BLIP** — найважча модель, але має такий самий розмір embedding, як CLIP (512).

**Висновок:** якщо потрібно мати багаті вектори — ResNet; якщо зображення + caption — BLIP, але це потребує ресурсів.

**📌 Загальний висновок**

| **Якщо потрібно...** | **То варто обрати...** |
| --- | --- |
| Швидко запускати і отримати embedding | **CLIP** або **ResNet** |
| Візуальний + текстовий опис (captioning) | **BLIP** |
| Потужний візуальний енкодер для downstream задач | **ViT** або **ResNet** |
| Мінімізувати RAM/CPU навантаження | **ResNet** |

[

{

"id": "...",

"title": "...",

"articles": [...],

"images": [

{

"url": "...",

"alt": "...",

"data": "...", # base64

"caption": "..."

},

...

],

...

},

...

]

**✅ Що саме підтверджено:**

* JSON тепер зчитується правильно як список об’єктів.
* Було знайдено 10 зображень із data (base64).
* Модель CLIP ViT-B/32 завантажена.
* Ембедінги згенеровані і збережені в data/image\_embeddings.json.
* Працює навіть приклад обчислення схожості.

**🧠 Якщо хочеш іти далі:**

Ось кілька наступних кроків, які можуть бути цікавими:

1. **Пошук схожих зображень**:
   * Побудуй API чи UI, де можна вводити зображення або alt-текст, а система показує схожі.
2. **Візуалізація ембедінгів**:
   * Використай t-SNE або UMAP для зменшення до 2D і побудови інтерактивної карти.
3. **Кластеризація**:
   * Використай KMeans, DBSCAN, тощо — щоб згрупувати схожі зображення.
4. **Інтеграція з текстовими ембедінгами**:
   * Зіставлення alt-тексту чи заголовків з ембедінгами зображень для аналізу узгодженості.

## Why Move to Qdrant?

**Current limitations:**

* All data must fit in RAM
* No persistence across sessions (beyond PKL cache)
* No advanced filtering or indexing
* Limited scalability
* No concurrent access

**Qdrant benefits:**

* Persistent storage
* Efficient vector indexing (HNSW)
* Advanced filtering capabilities
* Scalable and production-ready
* REST API for easy integration

args = parser.parse\_args([

"--qdrant-url", "https://2829a7bb-1713-4e6c-b1b1-d8ffbf38a124.europe-west3-0.gcp.cloud.qdrant.io",

"--qdrant-key", "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2Nlc3MiOiJtIn0.yR7PB8U3c1ctT4OGCGLQnqE7-SRYDeapPDmCrAOozek",

"--text-pkl", "embedding\_cache/text\_embeddings.pkl",

"--image-pkl", "embedding\_cache/image\_embeddings.pkl",

"--clear",

"--validate",

"--test"

])

args.qdrant\_url = "https://2829a7bb-1713-4e6c-b1b1-d8ffbf38a124.europe-west3-0.gcp.cloud.qdrant.io"

args.qdrant\_key = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2Nlc3MiOiJtIn0.yR7PB8U3c1ctT4OGCGLQnqE7-SRYDeapPDmCrAOozek"